NASA SP-5069

TECHNOLOGY UTILIZATION

MATHEMATICAL COMPUTER PROGRAMS

A COMPILATION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NASA SP-5069

MATHEMATICAL COMPUTER PROGRAMS

A COMPILATION

Technology Utilization Division
OFFICE OF TECHNOLOGY UTILIZATION 1966
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Washington, D. C.

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $1.00

Foreword

The Administrator of the National Aeronautics and Space Administration has established a technology utilization program for "the rapid dissemination of information... on technological developments... which appear to be useful for general industrial application." From a variety of sources, including NASA Research Centers and NASA contractors, space-related technology is collected and screened; and that which has potential industrial use is made generally available. Information from the Nation's space program is thus made available to American industry, including the latest developments in materials, processes, products, techniques, management systems, and analytical and design procedures.

This publication outlines several mathematical programs and programming techniques for digital computers which are available separately or as a collection through the NASA technology utilization program. Although the functions which these programs perform are not new and similar programs are available in many large computer center libraries, this collection of programs may be of use to centers with limited systems libraries and for instructional purposes for new computer operators. Most of these programs are part of the operating system at North American Aviation, Inc. and were written by W. Kane, S. Kory, W. Vinson, R. Mittleman, S. Deifik, T. Highwort, G. Murine, and R. Wright. The others were written by S. O. Moy (The Boeing Company), D. Franz and H. P. Mitchell (The Chrysler Corporation), William A. Mersman (Ames Research Center), and Dr. Erwin Fehlberg (George C. Marshall Space Flight Center).

This report was prepared by J. Arnold, J. Simpson, and T. Leth of Illinois Institute of Technology Research Institute.

The Director,
Technology Utilization Division
National Aeronautics and
Space Administration

Contents

Introduction

The computer programs described in this publication were, for the most part, run on IBM 7094 computers. Sufficient information is provided on each program to allow a prospective user to assess its usefulness. The compatibility of these programs with other equipment depends upon the programming language and operating system used, as well as type of machines available. Detailed descriptions of individual programs are available from the responsible Technology Utilization Offices, at the addresses indicated at the end of each description.

Programs described are available for purchase at a nominal cost (sufficient to cover duplication). Necessary information and forms for purchasing are included with responses to requests for detailed descriptions.

Section I — Functions

ARC SINE AND ARC COSINE FUNCTIONS

Program Description: This is a Fortran IV subroutine which computes either the arc sine or the arc cosine (angle A), as desired.

Numerical Method: The method of calculation of the arc sine and arc cosine depends on the value of the argument (sine or cosine) and is as follows:

Arc Sine:

a. $|\text{sine}| \leq 0.27424$; arc sine is computed using the standard infinite series which terminates when the terms become less than sine / 20 000 000.

b. $|\text{sine}| > 0.27424$; arc sine is computed using Taylor-series expansions.

Arc Cosine:

a. $|\text{cosine}| > 0.98$; arc cosine is computed using a four-term polynomial.

b. $|\text{cosine}| \leq 0.98$; arc cosine is computed using the arc sine function and subtracting the complement of the angle form $\pi/2$.

Range of Data:

Arc Sine: $-\pi/2 \leq A \leq +\pi/2$

Arc Cosine: $0 \leq A \leq \pi$

Limitations and Problems:

a. If the sine exceeds 1.00, the arc sine is set equal to $\pi/2$ and an error routine is called.

b. If the cosine exceeds 1.00, the arc cosine is set equal to 0 or π, whichever is appropriate, and an error routine is called.

Accuracy: In the Taylor-series expansion calculations, the accuracy is ±2 in the 8th significant figure.

Program Storage:

Deck Size: (Binary) 43 cards

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-340

ARC TANGENT FUNCTION

Program Description: This is a Fortran IV subroutine which computes the arc tangent (angle A) given the sine and cosine. The form and range in which the angle is presented may be specified. In addition to computing the arc tangent, the program also computes and prints the tangent of the angle.

Numerical Method: The program computes the arc tangent by first deriving the angle from the sine and cosine and then applying the standard library function for the arc tangent (Taylor series approximation). The program checks the compatibility of the sine and cosine by comparing $\sin^2 + \cos^2$ with 1.0; if the error is greater than 0.00001, an error routine is called.

Range of Data: The arc tangent may be specified to fall in one of the following ranges: $0 \leq A < 2\pi$; $-\pi < A \leq +\pi$; $0^o \leq A < 360^o$; or $-180^o < A \leq +180^o$.

Input/Output Format: The input data contains the sine and cosine plus an integer which specifies the output format. The program returns the tangent and arc tangent as output.

Limitations and Problems: If the cosine is zero, the tangent is returned as $\pm 10^{35}$.

Accuracy: The accuracy of this subroutine is determined by the accuracy of the library arc tangent routine.

Program Storage:

Deck Size: (Binary) 32 cards

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-341

ROOT EXTRACTION

Program Description: This is a Fortran II subroutine which determines the intersection of a particular function with the x-axis over a given range (determines the root of the function).

Numerical Method: The program examines the function starting from the left limit in increments determined by the input data. When a change of sign of the ordinate is detected, the program returns to the previous value of x, divides the increment by 10, and proceeds as before. When an exact root is found or one that differs by a specified smallest increment, the value of x (the root) and the functional value are printed.

Range of Data: Any single-argument function within the range of Fortran II variables.

Input/Output Format: The input data must specify the right and left limits of the range, the first search increment, the smallest increment, and the function.

Limitations and Problems: If the function is periodic and the search increment is chosen equal to the period, none of the axis intersections would be apparent.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-472

GAMMA FUNCTION

Program Description: This is a Fortran II subroutine which evaluates the Gamma function ($\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$).

Numerical Method: If the value of α is in the interval $2 \leq \alpha < 3$, the Gamma function is evaluated using the Chebyshev polynomial approximation formula:

$$\Gamma(2+x) = \sum_{v=0}^{n} A_v x^v \quad (0 \leq x < 1)$$

where A_v are coefficients which are stored within the program. When α does not lie within this interval, the formula:

$$\Gamma(x+1) = x\,\Gamma(x)$$

is applied to bring the function within the interval.

Range of Data: The maximum value of α is determined by the largest number the particular machine can store. For the IBM 7090, the maximum word size is 10^{38}, which limits the value of α to 34.

Input/Output Format: The input and output of this program are single-valued; the argument, α, is supplied as input and the value of the Gamma function is the output.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-338

Section II — Vector Operations

VECTOR OPERATIONS

Program Description: This program consists of four separate mutually independent subroutines which are coded in the MAP language for use on a Fortran IV system. The capabilities of the program include the following vector operations:

a. addition and subtraction of vectors
b. finding the magnitude of a vector
c. normalizing a vector
d. finding dot and cross products of two vectors
e. finding the cross products of a vector and three column vectors

Numerical Method: The following operations can be performed by this program:

1. $VC = VA + VB$
2. $VC = VA - VB$
3. $S = VA \cdot VB$
4. $S = |V|$
5. $U = \frac{V}{|V|}$
6. $S = |V| \; U \; \frac{V}{|V|}$
7. $VC = VA \times VB$
8. $VC1 = VA \times VB1$
   $VC2 = VA \times VB2$
   $VC3 = VA \times VB3$

where V, VA, VB, VC are vectors; A, B, are 3 x 3 matrices; V is the magnitude of V; VC1, VC2, VC3 are column vectors of C; VB1, VB2, VB3 are column vectors of B; VA · VB is the dot product of VA and VB, and VA x VB is the cross product of VA and VB.

Range of Data: This program will accept any data which is compatible with the Fortran IV language.

Input/Output Format: For each operation, the input format must be declared.

Limitations and Problems: One array of Block Common must be set up in the calling program with the name BLOCK 1. The minimum dimension of BLOCK 1 required by this program is 1.

Accuracy: All operations are performed in single precision.

Program Storage:

Core:	DOT PRD	20 decimal
	X PROD	40
	MAG33V	43
	VADSUM	36
	Total	139 decimal

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-330

RESOLVED VECTOR COMPONENT ARC TANGENT FUNCTION

Program Description: This is a Fortran II subroutine which computes the arc tangent (angle A) given the x and y components of a vector. The form and range in which the angle is presented may be specified. In addition to computing the arc tangent, the program also computes and prints the tangent of the angle.

Numerical Method: The program computes the arc tangent by first deriving the angle from the vector components and then applying the standard library function for the arc tangent (Taylor series approximation). If the value of x is 0 and y is non-zero, the arc tangent is assigned the value of $\pm\pi/2$, $\pm 90^{o}$, $3\pi/2$, or 270^{o} and the tangent is assigned the value of -10^{35}. For $|\tan A| \leq 0.00001$, values of π, 180^{o}, 2π, or 0^{o} are assigned depending on the range selected.

Range of Data: The arc tangent may be specified to fall in one of the following ranges: $0 \leq A < 2\pi$, $-\pi < A \leq +\pi$, $0^{o} \leq A < 360^{o}$, or $-180^{o} < A \leq 180^{o}$. The largest variable used must have a magnitude less than 10^{38}.

Input/Output Format: The input data contains the x and y components of the vector plus an integer which specifies the output format. The program returns the tangent and arc tangent as output.

Accuracy: The accuracy of this program is determined by the accuracy of the arc tangent library routine.

Program Storage:

Core: 226 decimal positions; 617 decimal positions including auxiliary routines

Deck Size: (Binary) 13 cards
(Source) 55 cards

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-339

Section III — Matrix Operations

EVALUATION OF A DETERMINANT

Program Description: This is a subroutine to evaluate an N x N determinant by first reducing it to a triangular matrix and then taking the product of the diagonal elements. The subroutine is written in Fortran II.

Numerical Method: The evaluation procedure is based on the algebraic theorem that "the determinant of a triangular matrix is the product of its diagonal entries."

Range of Data: The procedure accepts any single precision numbers compatible with the Fortran II system.

Input/Output Format: The subroutine requires the matrix A, and the number of rows, N. It returns the value of the determinant, D.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-354

GENERALIZED DIMENSION MATRIX OPERATIONS

Program Description: This is a routine for matrices of general (M x N) dimensions which is coded in MAP language for the Fortran IV system. The overall routine consists of four mutually independent subroutines which may be loaded individually or with any combinations. Each of the four subroutines has one entry point and performs the operation specified in the setup information.

The four subroutines are: inversion of an N x N matrix (INVNNM); multiplication of an M x N matrix by an N x P matrix (MTXMPY); finding the transpose of an N x N matrix (TRNPNN); and outputting an M x N matrix according to a format that varies with M (WRIMX).

Numerical Method: The inversion routine performs column interchanges as a means of improving accuracy, but the method of finding the inverse is the standard one of performing elementary row operations on the original matrix. The inversion subroutine incorporates a simple error indicator, IX, which takes on values less than or equal to zero indicating a possible error in the inverse or that any of the N divisors are zero.

Range of Data: The system accepts any single precision numbers compatible with the Fortran IV system.

Input/Output Format: The subroutines each require all pertinent dimensions of the input matrices and the output matrices created.

For the matrix output routine a maximum of seven numbers will be printed out per line; an E17.8 format is used.

Limitations and Problems: The inverse subroutine destroys the original matrix, storing the inverse in its place. The inverse routine also requires two arrays of block common whose size is determined by the size of the input matrix. The other routines require no block common.

Program Storage:

Core:	INVNNM	routine requires	173 decimal locations.
	MTXMPY	routine requires	60 decimal locations.
	TRNPNN	routine requires	38 decimal locations.
	WRIMX	routine requires	47 decimal locations.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-350

DOUBLE PRECISION MATRIX OPERATIONS

Program Description: This routine consists of two mutually independent subroutines for double precision operations on an M x N matrix. It is coded in MAP for the Fortran IV system. The two subroutines employed allow for (1) finding the inverse of an N x N matrix, or (2) multiplying two compatible matrices; the subroutines may be loaded independently or together.

The double precision format of this subroutine gives extra accuracy in the computed results; this is of special importance in the handling of large matrices.

Numerical Method: The inverse routine performs column interchanges as a means of improving accuracy. Aside from this, the inverse is found by performing elementary row operations on the original matrix. The inversion subroutines also incorporate a simple error indicator for noting either a possible computational error, or that any of the N divisors are zero.

Range of Data: The routine accepts any double precision numbers compatible with the Fortran IV system.

Input/Output Format: Each subroutine requires pertinent dimensions of the input and output subroutines.

Limitations and Problems: The inverse subroutine destroys the original matrix, storing the inverse matrix in its place. In addition, the inverse routine requires two arrays of block common of a size determined by the input matrix.

The multiplication subroutine does not require block common.

Program Storage:

Core: Source program storage required: 65 decimal locations for the multiplication subroutine, and 190 locations for the inversion routine.

Miscellaneous: This program is essentially identical to the corresponding two subroutines in the preceding program (UT 307) with the exception that this program uses double instead of single precision.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-329

SPECIALIZED DIMENSION MATRIX OPERATIONS

Program Description: This is a multi-purpose routine for 3 x 3 matrices. It is written in MAP for the Fortran IV system. This routine is designed to minimize storage space and faster routines are available. It is recommended that this program be used when storage is limited.

The following operations are supplied by this routine: (1) finding a matrix inverse, (2) re-orthogonalizing a near-orthogonal matrix, (3) transposing a matrix, (4) evaluating the determinant of a matrix, (5) moving a matrix from one array to another, (6) multiplying a matrix by another matrix or a column vector, and (7) multiplying a transposed matrix by a matrix or a column vector. These separate subroutines may be used individually or in combinations.

Numerical Method: The inverse is calculated by using the formula $A^{-1} = (\det A)^{-1}$ adj. A. An iteration procedure is used to converge the orthogonal matrix by the formula $B = A\ (\frac{3}{2}I - \frac{1}{2}A^{t}A)$. A and A^{t} are the original matrix and its transpose, respectively. A^{-1} is the inverse of the original matrix and B is the next approximation in the orthogonalization, and I is the identity matrix.

The iteration formula is tested for convergence by checking that the row norms of $A^{t}A - I$ are less than 1.0, this condition indicating convergence.

Range of Data: The routine accepts any single precision numbers compatible with the Fortran IV system.

Limitations and Problems: This routine is limited to use with 3 x 3 matrices. One array of Block Common must be set up in the calling routine with the name BLOCK 1; the minimum dimension of BLOCK 1 depends on the subroutine being used, but a minimum dimension of 19 will be adequate for all subroutines.

Accuracy: The matrix inverse incorporates an internal error analysis to determine the number of places of significance of the inverse. The orthogonalization iteration proceeds until the row norms of $A^{t}A-I$ have converged to .000001. If this condition does not obtain it is indicated by a control variable.

Program Storage:

Core: The program requires 356 decimal locations.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-328

LARGE MATRIX MULTIPLICATION ROUTINE

Program Description: This is a special program developed to circumvent core storage problems in multiplying conformable matrices of large size. It has been used in the multiplication of matrices as large as 100 x 100. This program is written in Fortran II, and is intended for use in situations where limited storage does not allow enough locations to store the product matrix; this matrix is then stored into one of the original matrices. Either of the original matrices can be chosen by the user as storage locations for the product matrix.

Range of Data: The subroutine accepts all single precision numbers compatible with the Fortran II system.

Input/Output Format: Specifications of size must be made for all matrices involved.

Limitations and Problems: Given matrix A of size M x N and matrix C of size N x L, then if the product matrix is to be stored in A the number of columns in A must be greater than or equal to L. Similarly, the number of rows in B must be greater than or equal to M if B is chosen as storage for the product matrix.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-335

MATRIX INVERSION ROUTINE (MAINV)

Program Description: This is a specialized subroutine for finding the inverse of a non-singular N x N matrix. The original matrix is not destroyed and its calculated inverse is stored in additional locations by the subroutine. The subroutine is written in Fortran II.

Numerical Method: The inversion is performed iteratively by reducing the original matrix to an identity matrix by a sequence of row operations and then applying the same operations to the identity matrix to generate the inverse.

Range of Data: This subroutine accepts any single precision numbers compatible with the Fortran II system.

Limitations and Problems: The program assumes that the original matrix is non-singular and that all of its diagonal elements are non-zero.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-362

MATRIX REORTHOGONALIZATION ROUTINE

Program Description: This is a special purpose subroutine, programmed in Fortran II, for producing an orthogonal 3 x 3 matrix from one which is near orthogonal.

Numerical Method: The orthogonalization is performed by an iteration process using the formula $V = U(\frac{3}{2} I - 1/2\ U^t U)$. U and U^t are the matrix and its transpose, respectively; I is the identity matrix; and V is the next approximation.

The process is tested for convergence by checking that the row norms of $U^t U - I$ are less than 1.0, this condition indicating convergence.

Range of Data: This subroutine accepts any single precision numbers compatible with the Fortran II system.

Limitations and Problems: The original matrix is destroyed and its inverse is stored in its place.

Accuracy: The subroutine also supplies a control variable, KOUNT, whose value indicates whether the matrix was orthogonalized to the required accuracy, whether it could not be orthogonalized, or whether the maximum number of iterations was exceeded.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-351

Section IV — Integration

INTEGRATION ROUTINE

Program Description: This program provides a small fixed step integration package to permit the integration of tabular derivatives without a high degree of accuracy. Stops can be made on integrated or non-integrated variables or on the independent variable. No interpolation is made to provide values between interval endpoints. The case ends with the printing of the first interval endpoint beyond the stop value.

Subroutines NTEGRT, MOVEX, and MONINT are written in MAP for a Fortran IV system. NTEGRT and MOVEX are essential to the package. NONINT is optional and used only when there are stops on non-integrated variables. NTEGRT does the actual integration. MOVEX updates the indicators for the next cycle and checks for stops on the independent variable when both corrected ordinates and derivatives are available.

Numerical Method: A two-point predictor-corrector system of numerical integration is used by NTEGRT for the actual integration. The predictor equation is $Y_{n+1} = Y_n + DX\,(\frac{3}{2}Y'_n - \frac{1}{2}Y'_{n-1})$. The corrector equation is $Y_{n+1} = Y_n + DX\,(\frac{1}{2}Y'_{n+1} + \frac{1}{2}Y'_n)$.

Range of Data: This program will accept any data compatible with the MAP language.

Input/Output Format: Input data must include (1) initial value of the independent variable, (2) the step size, (3) the print interval, and (4) the initial values of all integrated variables.

Limitations and Problems: An array of ordinate values obtained by integration and the derivative which is integrated to give the ordinate values requires a dimension of 7. An array of values for the non-integrated variable requires a dimension of 4. A block of labeled common must be set up with the name, VARX, and must contain 10 locations.

The section which evaluates the derivatives must begin with a statement number. A print should be made only when both corrected ordinates and derivatives are available. If a print is made, L(3) should also be tested to see if a stage or case ending has been found. NTEGRT should be called for each variable integrated with the ordinate as the argument. Following the last "call NTEGRT" statement, the program should call MOVEX. The program should then transfer control to the derivative section.

Program Storage:

Core:	NTEGRT	113 decimal	Binary Deck Size:	NTEGRT	12 cards
	MOVEX	39		MOVEX	8
	NONINT	28		NONINT	7
	Total	180 decimal		Total	27 cards

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-336

INTEGRATION ROUTINE

Program Description: This program integrates a set of differential equations using Fortran II language. It consists of the following five subroutines:

a. PRECOR, which merely sets certain constants which are needed by the program but which may be altered by input data.

b. SETUP, which initializes all counters and indicators and determines the minimum computing interval.

c. INTEG, which actually integrates the differential equation and evaluates the error associated with the integration.

d. UPDATE, which updates all variables, keeps track of the print cycle, examines the error indicator, and alters the computing interval.

e. ENDIT, which checks for stop conditions and interpolates for the final interval of integration.

Numerical Method: A fourth order Runge-Kutta formula is used to start the integration or to calculate a point subsequent to reducing the interval size. An Adams-Moulton Predictor-Corrector technique is then used to complete the integration.

Either a fixed or variable interval mode may be employed. The variable mode is recommended, since this allows the step size to be decreased, insuring accuracy, or increased, giving efficiency.

Both multiplicative and additive round down errors are minimized in this program.

A cutoff routine allows for a stop on the independent variable, the integrated variable, their derivatives, or non-integrated variable computed elsewhere.

Range of Data: This routine will accept any numbers compatible with the Fortran II language.

Input/Output Format: Six variables must be declared.

Accuracy: Integration error limits:

minimum error 0.00001
maximum error 0.001

Program Storage:

Core: This program occupies 100 cells of common.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-483

INTEGRATION ROUTINE

Program Description: This is a highly efficient routine for numerical integration written in MAP for the Fortran IV system. This program incorporates a set of starter equations of higher accuracy than in many integration routines and a highly efficient procedure for meeting specified ending conditions. This results in a routine which minimizes computer time while maintaining a high degree of accuracy.

The subroutine has three entry points, any of which may be used in a given program. The subroutine also has internal procedures to allow stopping at (1) a value specified for the independent variable, (2) a value of any ordinate obtained by the integration, or (3) a specified value of any non-integrated variable defined by the main program. A counter shows how many of the stopping conditions have been met.

Numerical Method: The routine has a built-in starting procedure which uses a series of predictor-corrector formulas similar to (and in some cases identical to) the Newton-Cotes quadrature formulas. The interval of integration is variable and it is increased or decreased automatically by the subroutine. The variable step size gives the shortest possible machine time for the calculation while the accuracy is maintained at a level compatible with the accuracy specified by the input to the routine. The integration can go either forwards or backwards in terms of the independent variable, as set by the input data from the calling program.

Range of Data: This routine will accept any single precision numbers compatible with the Fortran IV system.

Input/Output Format: An array of values for the independent variable, an array of values for the ordinate obtained by integration, and an array of values for the derivative which are integrated to give the ordinate values must be specified in the calling statement. Additional arguments required in the calling statement depend on the entry point used.

Limitations and Problems: Only one of the entry points to the subroutine may be used in any one program.

Program Storage:

Core: The maximum storage required for the program is 2249 decimal locations, which could be reduced to 1982 by removal of one of the subsidiary subroutines. The complete extra-print deck required 3176 and 2799 decimal locations respectively.

Deck Size: 153 cards for the production deck and 200 for the extra-print deck.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Manned Spacecraft Center
Houston, Texas 77058
Reference: MSC-479

ADAMS-MOULTON INTEGRATION SUBROUTINE (AMINT)

Program Description: This subroutine will numerically integrate a set of N simultaneous first order ordinary differential equations, using either of the three following methods: (1) Adams-Moulton method with a fixed step size, (2) Adams-Moulton method with a variable step size, or (3) a fourth order Runge-Kutta method with a fixed step size.

Numerical Method: The system of simultaneous differential equations of the first order must be in the following form:

$$Y_1' = f_1\ (t,\ Y_1, \ldots Y_N)$$

$$Y_2' = f_2\ (t,\ Y_1, \ldots Y_N)$$

$$Y_i' = f_i\ (t,\ Y_1, \ldots Y_N)$$

$$Y_N' = f_N\ )t,\ Y_1, \ldots Y_N)$$

where t is the independent variable and $Y_1, \ldots Y_N$ are the dependent variables.

A higher order differential equation, or a system of equations including some high order members, may be reduced to a set of first-order equations by making a simple change of variable. An nth order equation,

$$Y^{(n)} = f\ (X,\ Y,\ Y',\ Y'',\ \ldots,\ Y^{(n-1)}\ )$$

may be transformed by letting

$$Y = Y_0,\ Y' = Y_1,\ Y'' = Y_1' = Y_2,\ Y''' = Y_1'' = Y_3, \ldots\ldots\ \ldots\ldots$$

$$\ldots Y^{(n)} = \ldots = Y_{n-2}'\ f\ (X_1,\ Y_0,\ Y,\ Y_2,\ \ldots Y_{n-2})$$

Such a simultaneous system can be handled by this subroutine.

Range of Data: This subroutine will accept any single precision numbers compatible with the Fortran IV system.

Input/Output Format: The user must provide the N first order derivatives and the dependent variables must be determined for a given t.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Marshall Space Flight Center
Huntsville, Alabama 35812
Reference: M-FS-465

SELF-STARTING METHODS TO NUMERICALLY INTEGRATE ORDINARY DIFFERENTIAL EQUATIONS

Description: These are methods used to devise a self-starting, multistep procedure for the numerical integration of ordinary differential equations. The classical, multistep, predictor-corrector procedures for the numerical solution of systems of ordinary differential equations are generalized to provide compatible, self-starting methods that produce all the required backward differences directly from the initial equations. Explicit algorithms and tables of numerical coefficients are given for starting and continuing the numerical integration of the equations.

Most multistep methods are not self-starting, and single step methods such as that of Runge-Kutta are used to obtain starting values for the integration. This requires nonessential tallying to determine whether enough starting values have been obtained.

Numerical Method: The general problem is to devise algorithms for calculating x_n, y_n, and f_n [where $x_n = x(t_n)$, $y_n = y(t_n)$, and $f_n = f(x_n, y_n, t_n)$] for $n = 1, 2, 3 \ldots$ given the differential equations $dx/dt = y$ and $dy/dt = f(x, y, t)$ and the initial values $x_0 = x(t_0)$ and $y_0 = y(t_0)$. The theory for first order systems is obtained by ignoring x in these equations. The procedure used is the conventional one of approximating the function (f) by a polynomial t of degree q.

In order to achieve the best compromise between the requirements of speed, accuracy, and programming compactness, the following procedures are used in the integration:

I. Fourth order methods are used for first order equations ($q = 4$) and sixth order methods are used for second order equations ($q = 6$).

II. The iterated starter, which initializes the algorithms and then iterates the single set of equations, is used and iterated eight times. The iterated starter is superior to a "bootstrap" starter (essentially an efficient way of obtaining first approximations in the right hand side of the algorithms) because the bootstrap starter, although efficient in practice, is awkward and space-consuming when programmed for automatic computers due to the multiplicity of algorithms and matrices required.

III. The summed form of the predictor-corrector algorithm is used in backward difference form. The effectiveness of the summed form of the predictor-corrector formula has long been known to astronomers. The use of backward differences in forward integration is preferable to the use of backward ordinates for two reasons: (1) the backward ordinate formula tends to add nearly equal quantities of alternating sign, whereas the backward difference formula adds monotonically decreasing quantities; and (2) the availability of the difference table makes error estimation and automatic adjustment of the interval size a straightforward procedure.

IV. Four extra significant decimal digits are carried, in floating-point form, to control round-off errors.

Miscellaneous: A complete report on the theory and use of the self-starting procedure for differential equations is available in NASA Tech Note TN D 2936 entitled "Self-Starting Multistep Methods for the Numerical Integration of Ordinary Differential Equations."

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Ames Research Center
Moffett Field, California 94035
Reference: ARC-50

NEW HIGH-ORDER RUNGE-KUTTA FORMULAS WITH STEP SIZE CONTROL FOR SYSTEMS OF FIRST- AND SECOND-ORDER DIFFERENTIAL EQUATIONS

Description: This is a mathematical method to be used when the numerical solution (by electronic computer) of a differential equation requires changing the integration step size. This method involves the use of a new set of Runge-Kutta type formulas which allows solutions of any desired accuracy by repeated differentiation of the equations. The simpler high-accuracy methods (e.g., Gauss) can no longer be used in situations where the step size must be changed, and Runge-Kutta type formulas are very convenient since no knowledge of previous steps of the differential equation or its solution need be carried forward. However, the widely used Runge-Kutta fourth-order method is not of high accuracy and therefore requires small step sizes which increases computing time greatly. Furthermore, several time-consuming evaluations of the first derivative are needed and the method does not provide an estimation of truncation errors.

Numerical Method: Solutions which are correct up to the Taylor series term of order h^{m+4} (where h is the integration step size) can be obtained by an m-fold differentiation of a system of first- or second-order differential equations, and a very convenient simplified transformation of the system. These require three evaluations of the differential equations. Two additional evaluations yield formulas of the (m+5)th order. The differences between the two formulas can be used on a computer to set up a step size control procedure for the (m+4)th order formulas. For second-order systems only one additional evaluation is required for the step size control (instead of two), if step size control tests for the first derivatives are disregarded.

Miscellaneous: The new Runge-Kutta type formulas, their transformation, and the application of this technique to the solution of a heavy asymmetrical top problem are presented in a report entitled "New High-Order Runge-Kutta Formulas with Step Size Control for Systems of First- and Second-Order Differential Equations" by Dr. Erwin Fehlberg. The execution of the required differentiations of the differential equations was explained in an earlier paper (entitled "Runge-Kutta Type Formulas of High-Order Accuracy and Their Application to the Numerical Integration of the Restricted Problem of Three Bodies"). This paper introduced the system and showed that such a differentiation can easily be performed on the computer by means of recurrence formulas if, by introducing auxiliary functions, the differential equations can be transformed into an algebraic system of the second degree.

In addition to the two reports mentioned, several programs are available which were used to test the method on a computer. These formulas required less than 10 percent of the computer time necessary for other Runge-Kutta solutions when tested on an IBM 7090 computer using double precision and automatic step size control.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Marshall Space Flight Center
Huntsville, Alabama 35812
Reference: M-FS-424 & 412

Section V — General

ANALYTIC NTH ORDER AND PARTIAL DIFFERENTIATION OF ALGEBRAIC AND TRANSCENDENTAL EXPRESSIONS

Program Description: This is a program written in Fortran II and FAP which will generate the derivatives of a given class of algebraic and transcendental functions. The elementary functions are first transformed in such a way that their derivatives can be obtained simply by successive applications of the basic rules of differentiation. This program has proven itself useful especially where multiple order differentiation and partial differentiation of involved expressions are desired. Once the order of differentiation is set up any given number of expressions may be sequentially differentiated.

Beyond the primary function of performing analytical differentiation of expressions, this program has the following features:

1. The Ershov algorithm on which the program is based can be employed to facilitate many types of symbol manipulation processes.

2. With successive passes over the matrices formed by the transformations, higher derivatives and partial derivatives of expressions may be obtained.

3. Chain differentiation of functions such as $w = f(u, v)$, where $u = g(x, y)$ and $v = h(x, y)$ can be incorporated into the program.

4. An additional stage can be added which would transform the second set of triples into a machine language program which could be executed on the computer and would evaluate the derivative for given values.

5. A differential operator can be added to the algebraic language, thus permitting the translator to generate the required expressions for the derivatives and the coding necessary to evaluate these derivatives during the evaluation of high-order derivatives of complicated expressions.

Numerical Method: All expressions are assumed to be written in a simplified, algebraic compiler-like language such as Fortran. The expressions are augmented with a set of symbols used only within the algorithms. These are classed as operators and are assigned a precedence level.

The first step in the differentiation procedure is the transformation of the input expression into a table of triples which is equivalent to the Polish-prefix or parenthesis-free form of the expression. This transformation is accomplished by an algorithm developed by Ershov. The algorithm scans the expression from right to left. It alternately encounters an operand, then an operator. These elements are first transferred to an intermediate list, designated as the L-list. The entries of this list also alternate between operand and operator.

If the precedence of the current operator is less than the precedence of the previous operator, then the triple consisting of the last three entries on the L-list (operand-operator-operand) are removed from this list and transferred to the table of triples. This table of triples has been designated as the M-matrix for notational purposes. At the same time, an operand which represents the triple and specifies its row position in the M-matrix is added to the L-list. This procedure is continued until the entire expression has been processed and is transformed into the desired table of triples.

The next step in the differentiation is the development of a new table of triples which represents the derivative of the original expression in the same manner that the M-matrix represents the original expression. This new table is designated as the D-matrix. A set of equations representing the differentiation of elementary algebraic and transcendental functions completely determines the algorithm for developing the derivative of a single row of the M-matrix. The D-matrix can be constructed by beginning at the first row of the M-matrix and working downward, differentiating each row of the M-matrix order. In order to differentiate rows in the M-matrix that contain references to previous rows, information as to which row of the D-matrix represents the derivative of the i-th line of the M-matrix must be available. This information is stored in an auxiliary table, designated as the Q-table.

When several rows are added to the D-matrix as the parenthesis-free form of a compound expression of a given derivative, it is necessary that these rows be added in the same order that would occur should the Ershov algorithm be used to transform the derivative expression into the tabular form. This process facilitates the formulation of a consistent algorithm for transforming the D-matrix back into a parenthesized expression for output. Redundances which occur in generating the D-matrix can be eliminated in two ways. First, a special pass may be made over the D-matrix after it is generated to remove the redundances. Or, secondly, the removal of redundances can be incorporated into the algorithms for the generation of the D-matrix.

The final step in the differentiation program is the inverse of the initial step. Here, the parenthesis-free tabular form of the derivative is transformed into a parenthesized string or expression.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Marshall Space Flight Center
Huntsville, Alabama 35812
Reference: M-FS-140

FREQUENCY AND TIME RESPONSE PACKAGE

Program Description: This program forms a polynomial fraction by multiplying polynomials and adding together the products to successively form a numerator and denominator. Either the denominator, or both the numerator and denominator roots may be obtained. Frequency response may be obtained in any or all of three options:

1. for values of j-omega between a lower and upper limit, with phase shift between successive points controlled between specified tolerances;
2. for values of j-omega between a lower and upper limit, with a specified increment of j-omega between successive evaluations; and
3. for particular values of j-omega.

The polynomial fraction may also be evaluated for specified complex values of the argument. The inverse Laplace transformation of the polynomial fraction (time response) may be evaluated within a specified interval of time, with a specified increment of time between successive evaluations.

This program has been designed for economy of input and any number of evaluation runs may follow a polynomial input run.

Range of Data: The program will accept any single precision numbers compatible with the Fortran II system.

Input/Output Format: The input polynomials must be of degree not greater than 20. The degree of the computed numerator or denominator polynomial must not exceed 50. A maximum of 500 polynomials may be used as input to the program at any one time.

Limitations and Problems: The number of input points permitted for either the third frequency response option or complex evaluation option cannot be greater than 400. An algebra cannot contain more than 360 characters. A maximum of 500 polynomials may be used as input to the program at any one time.

Miscellaneous: The timing for this program is variable. However, roots and the first option of frequency response for 21 successive runs on transfer functions of moderate complexity (numerator and denominator of degree about 20) have been obtained in less than five minutes.

The source languages for this program are Fortran I and FAP for the IBM 7094 computer and operated under the IBSYS monitor.

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Marshall Space Flight Center
Huntsville, Alabama 35812
Reference: M-FS-508

GAIN AND PHASE OF TRANSFER FUNCTION AND MATRIX CURVE FIT

Program Description: This is a Fortran IV program which computes the transfer function by using a complex matrix curve-fit routine. The transfer function is used to compute its Gain and Phase in order to check the accuracy of the curve fit. It is also possible to input the transfer function and compute the Gain and Phase. The program allows the calculation of proper roots, pure real, pure imaginary, or complex conjugates. An additional modification has been made to correct shifts caused by unstable, complex roots.

Numerical Method: If the input is of a form other than root or polynomial, the program will compute the transfer function by using a double-precision complex curve fit.

Originally, the program utilized a single-precision complex curve fit. This was found unsatisfactory for higher order curves due to the existence of comparatively large imaginary portions of the calculated transfer function coefficients which, if correctly input, should theoretically have been zero. This curve fit error was compounded upon entrance to a routine utilized to determine the roots of the curve fitted by the single-precision complex matrix curve fit. The roots, instead of being pure real, pure imaginary, or complex conjugates, were found to be complex numbers and imperfect complex conjugates. These roots were then used to calculate the gain and phase of the transfer function. The incorporation of the double-precision complex curve fit has greatly increased the accuracy of the curve fit, and, as an additional precaution, the complex portions of the transfer function coefficients have been set to zero on entrance to the root extraction routine.

Range of Data: This program will accept any double-precision numbers compatible with the Fortran IV system.

Input/Output Format: The program will accept input data in root or polynomial form, in the form of gain and phase, gain d.b. and phase, or GR and GI and ω (omega).

Inquiries concerning this program may be directed to:

Technology Utilization Officer
Marshall Space Flight Center
Huntsville, Alabama 35812
Reference: M-FS-507

NASA-Langley, 1967